Navigator

NAVO MSRC

FALL 2003

Delivering Advanced Technology to the WARFIGHTER!

News and information from...
The Naval Oceanographic Office Major Shared Resource Center

1. REPORT DATE **2003**	2. REPORT TYPE	3. DATES COVERED **00-00-2003 to 00-00-2003**

4. TITLE AND SUBTITLE **NAVO MSRC Navigator. Fall 2003**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Oceanographic Office (NAVO),Major Shared Resource Center (MSRC),1002 Balch Boulevard,Stennis Space Center,MS,39522**	8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**28**	

The Director's Corner

Steve Adamec, NAVO MSRC Director

NAVO MSRC Supports Operation Iraqi Freedom

As one of four Major Shared Resource Centers (MSRCs) within the Department of Defense (DoD) High Performance Computing Modernization Program (HPCMP), the NAVO MSRC is a premiere provider of HPC services and support to Research and Development (R&D) communities throughout DoD. Here at the NAVO MSRC, a unique and critical portion of our mission is the provision of these same cutting-edge HPC resources directly to operational DoD users, i.e., warfighters.

One part of our technical mission is data storage and management. While this may seem to be a minor mission, the enormous amounts of data we accumulate, store, and disseminate to users can be essential in the time-critical planning and execution of DoD operations. For example, both computed and collected data from a variety of sources were stored and analyzed within the MSRC by NAVOCEANO personnel to determine how best to de-mine the Shatt al Arab straits and to determine the most appropriate shipping channels to allow supply ships to bring needed military equipment and humanitarian materials quickly into Iraq.

Using MSRC resources, NAVOCEANO personnel also provided critical support to the Navy, DoD Special Operations Forces, and to Army and Marine Corps forces on the ground in Iraq. Examples of the support provided to the forces of Iraqi Freedom include:

☞ High-priority development and execution of coastal ocean forecast models to describe the tidal forces and currents that might impact special operations activities.

☞ Identification, by NAVOCEANO Riverine Analysis Team members, of potential water hazards in the maze of Iraqi rivers and canals that range from unidentified rapids to soft riverbanks.

☞ Analysis by NAVOCEANO personnel of potential flood impacts from threatened Iraqi demolition of dams along the approaches to Baghdad. Using the MSRC HPC resources, they determined where the water would go, how long it would be present, and how coalition forces could bypass it.

High priority use of the HPCMP assets at NAVOCEANO for time-critical operational support has proven to be hugely beneficial for DoD operations and for the taxpayer. The HPCMP R&D community also clearly benefits from the enhanced resilience and availability of the HPC environment that the MSRC maintains to meet operational computing requirements.

In the future, the mission of the NAVO MSRC will undoubtedly continue to expand to meet the unique requirements of the DoD operations community. Here at the NAVO MSRC, we are proud to enable HPC to play an important part in maximizing America's Sea Power by applying relevant oceanographic knowledge across the full spectrum of warfare.

Contents

SALMON
Virtual Environment

The NAVO MSRC provides tailored support to the Climate, Weather, and Ocean (CWO) modeling community. This includes support in the realm of High Performance Computing (HPC), storage, and visual analysis. In the area of visual analysis, the NAVO MSRC Visualization Center has become proficient in developing interactive analysis applications, which allow users to interactively analyze and interrogate their full-resolution model output. These custom applications are efficient and portable and can usually be run on the users' desktops.
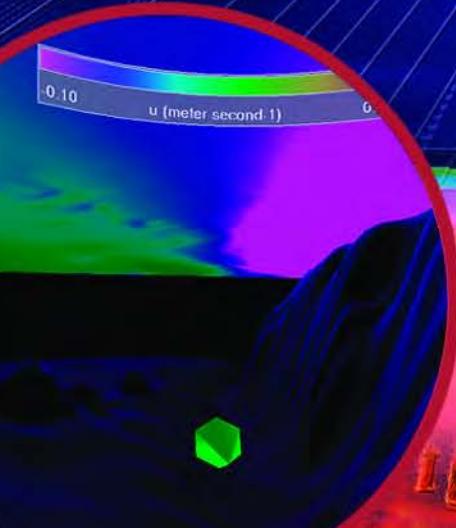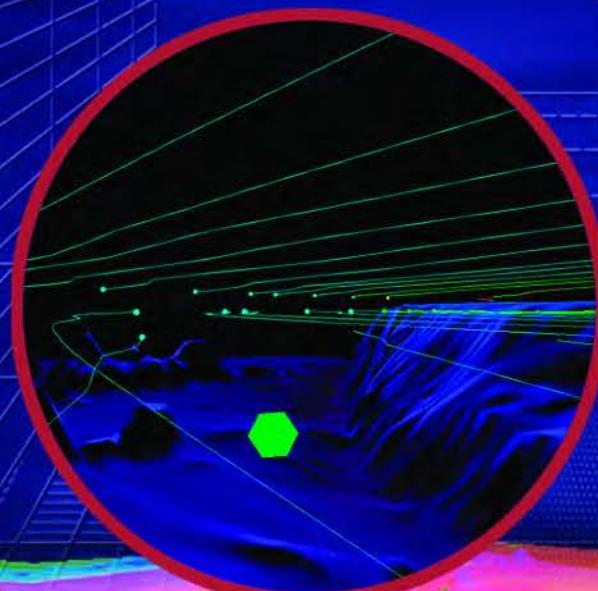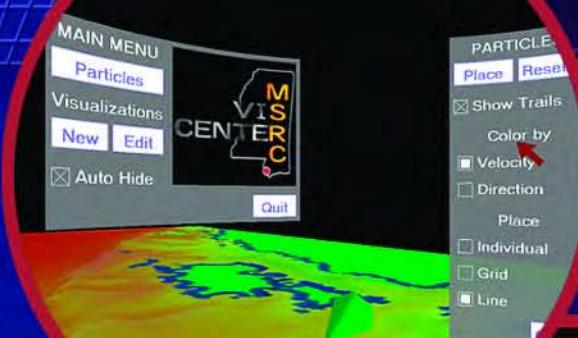
Multi-Level Parallel Paradigms for Flow-Induced Vibrations

Suchuan Dong, Didier Lucor, and George Em Karniadakis, Division of Applied Mathematics, Brown University, RI

Flow past a flexible cylinder subject to Vortex-Induced Vibrations (VIV) arises in numerous industrial and marine applications, for example, the flexible risers and tendons in petroleum production and marine tow cables (see Figure 1). Direct Numerical Simulations (DNS) have proven to be an effective tool to model such flows in both laminar and turbulent regimes. [1, 2] Realistic VIV simulations of cylinder flows require a large number of Fourier modes along the cylinder span and high resolutions in the streamwise and cross-flow directions. Parallel computations employing a single-level parallelism for this type of problems have clear performance limitations that preclude effective scaling to the larger processor count on modern supercomputers.

Careful analysis indicates that VIV computations demonstrate inherent hierarchical structures when the problem is discretized with spectral element methods. Consider an incompressible flow past a flexible cylinder subject to VIV. It is assumed the flow and cylinder variables are periodic in the homogeneous direction. A combined spectral element—Fourier discretization—can be employed to accommodate the requirements of high-order as well as efficient handling of multiply connected computational domain in the non-homogeneous planes. Spectral expansions in the homogeneous direction employ Fourier modes that are decoupled except in the nonlinear terms. Each Fourier mode is in a Two-Dimensional (2D) field and can be solved with the spectral element approach.

Computations on the Fourier modes, spectral element plane, spectral elements within the plane, and at the sub-element level form the hierarchy of operations in the solution process of the VIV problem. Multilevel parallelism naturally suits such VIV computations with inherent hierarchical structures.

MULTI-LEVEL PARALLEL PARADIGMS FOR VIV

This article presents two multilevel parallel paradigms for VIV based on Message Passing Interface/MPI/MPI and MPI/OpenMP, respectively. Figure 2a illustrates the MPI/MPI two-level parallel paradigm. The flow domain is first decomposed along the cylinder. Each sub-domain consists of one or more Fourier modes in the Fourier space.

At the first level are groups of MPI processes, with each group computing one sub-domain. The non-homogeneous spectral element planes are further decomposed at the second level. Each of the sub-domains at the second level comprises structured or unstructured elements. Accordingly, each MPI process within the group computes one sub-domain at the second level. Distinct communication characteristics manifest at the two levels and dominate different stages of computation.

The dominant pattern at the first level is the all-to-all communications for transposing the distributed matrices when the Fast Fourier Transformation
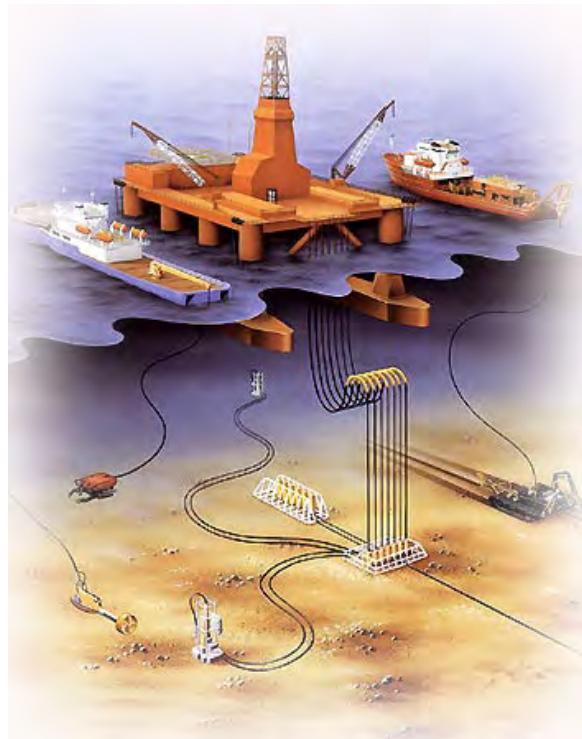
Figure 1. Flexible Risers.

(FFT) is evaluated in the non-linear terms and the velocity divergence. At the second level, reduction operations dominate the communications when the inner products are evaluated in a conjugate gradient iterative solver for computing the pressure and velocity. The communications on these two levels take place at different stages of the computation and alternate as the simulation marches in time.

Figure 2b provides a schematic for the MPI/OpenMP hybrid paradigm. The flow domain is decomposed in the homogeneous direction again. At the first level multiple MPI processes are employed, with each process computing one sub-domain. At the

second level, multiple OpenMP threads are employed to conduct the computations within each sub-domain in parallel.

Data exchange across sub-domains is handled with MPI. Within the sub-domain (or MPI process), accesses to shared objects by multiple threads are coordinated with OpenMP thread synchronizations. A coarse grain approach to OpenMP shared memory parallelism is taken, which reduces the OpenMP synchronizations significantly.[3,4]

MPI calls are handled by only one thread within each sub-domain. This configuration assembles the nodal messages into a single large message and thus reduces the network latency overhead. OpenMP barriers are the main type of synchronizations involved. The majority of barriers occur at the switching points between global and local operations. The authors have developed a consistent workload-splitting scheme for local and global operations that completely eliminates such barriers.[3]

RESULTS

In the research described in this article, these multilevel parallelization models were applied to VIV simulations of the flow past a cylinder. The first case is the turbulent flow past a stationary circular cylinder at Reynolds Number (Re) = 3900 based on the inflow velocity and the cylinder diameter. The MPI/OpenMP hybrid parallel paradigm was employed for this problem.

In the homogeneous direction there are 32 Fourier modes, and 32 MPI processes are employed in the computations so that each process computes one Fourier mode. P-type refinement is performed systematically in the x-y planes with the polynomial order increasing from 8 to 16 in all the elements. To demonstrate the effect of multi-threading, the wall clock time/step histories for all the polynomial orders are collected. The data with a single thread per MPI process are first collected. Then, as the polynomial order is increased, the number of OpenMP threads per process is increased approximately in proportion to the cost

M	K	P	DOF (million)	Processors	Wall time/step (s)
2	6272	5	2.1	16	0.83
8	6272	5	8.5	64	1.04
16	6272	5	17	128	1.15
32	6272	5	33.9	256	1.21
64	6272	5	67.7	512	1.54
64	6272	6	97.5	512	1.62
64	6272	8	173.4	512	3.31

Table 1. Wall clock timing versus problem sizes for cylinder flow at Re = 10,000. DOF: total degrees of freedom of the system. M: number of Fourier modes; K: number of spectral elements; P: polynomial order of elements. Fourier de-aliasing is employed in homogeneous z direction, and polynomial de-aliasing is employed in x-y planes.

increase in the single-thread case. Figure 3 shows the timing history as a function of the time step for single- and multiple thread cases. With a single thread per process, the wall clock time increases from about 4 to about 30 seconds. As the number of threads per process increases approximately in proportion, the wall clock time per step decreases dramatically and essentially remains constant for all the polynomial orders.

In the second case MPI/MPI two-level parallel simulations were conducted of the turbulent flow past a circular cylinder at the Reynolds number Re = 10,000. The number of Fourier modes in the homogeneous direction is varied between 2 and 64. At the first level, MPI processes are divided into groups (first level) based on the number of Fourier modes such that each group computes one Fourier mode.
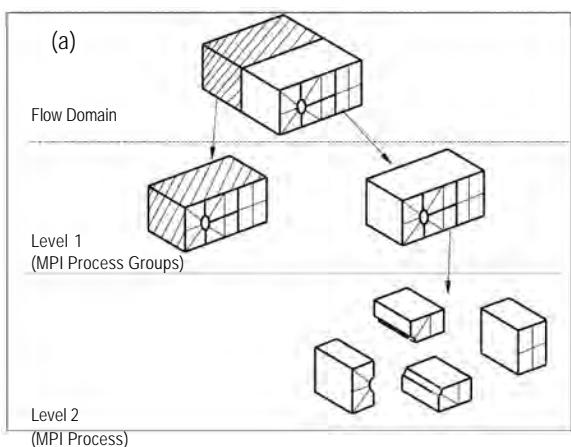
Figure 2. Schematics of multilevel parallelization models: (a) MPI/MPI and (b) MPI/OpenMP. In MPI/MPI, flow domain is first decomposed in the homogeneous direction (first level)—each non-homogeneous spectral element plane is further decomposed subsequently (second level). In MPI/OpenMP, flow domain is decomposed in the homogeneous direction maps onto MPI processes —multiple OpenMP threads within each MPI process compute sections of data in parallel.
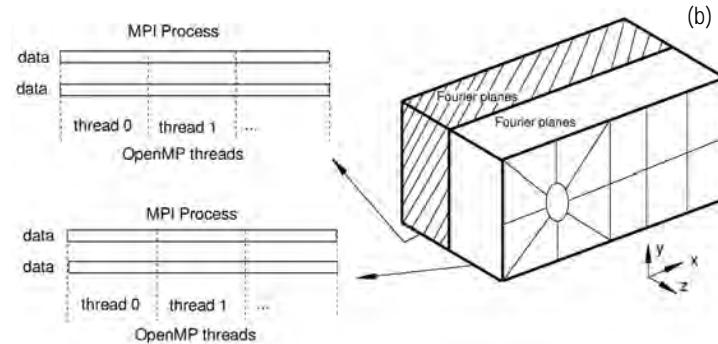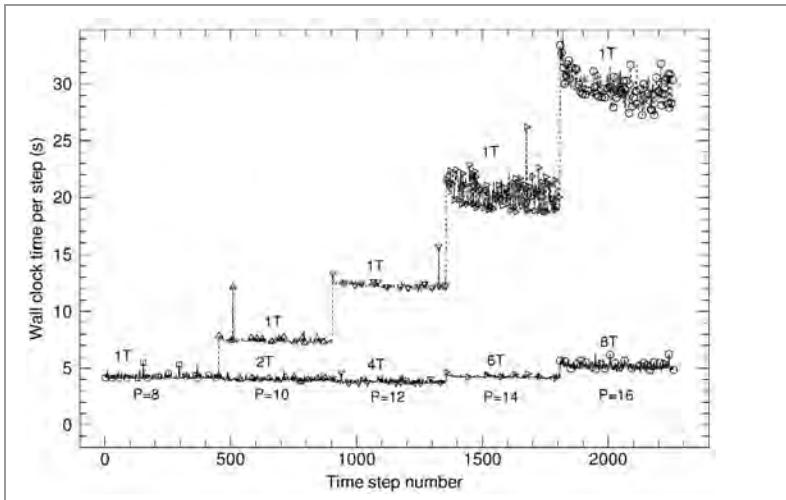
At the second level, eight processes are deployed in each group for this problem. Table 1 lists the wall clock timings for various grid resolutions. The capability of the multilevel parallel paradigm is clearly demonstrated by the large problem sizes (about 170 million degrees of freedom) and the performance (a few seconds per time step). The data also demonstrate the good scalability of this model. The drag coefficient computed from simulations is $C_D = 1.114$, which is in good agreement with experimental value $C_D = 1.186$.[5]

The third case simulates the turbulent flow past a rigid cylinder subject to vortex-induced vibrations at the Reynolds number Re = 1000 employing 32 Fourier modes in the homogeneous direction. The cylinder is allowed to move freely only in the transverse direction. Figure 4a shows plots of isocontours of the instantaneous pressure in the near-wake, which demonstrates the three-dimensionality of the flow structures. Figure 4b shows spatial-temporal contours of the drag (top plot) and lift (bottom plot) on the cylinder. Significant variations of the instantaneous lift and drag forces along the cylinder are observed.

Analysis indicates that regions with high lift amplitude correspond to situations where the cylinder displacement and the lift force are in phase, while regions with low lift amplitude correspond to situations where cylinder displacement and the lift force are out of phase. Indeed, strong three-dimensionality in the wake (See Figure 4a) influences the

topology of the vortices inducing forces that are not in phase with each other.

In real-world flow situations, a variety of uncertainties exist in the flow problems regarding the inflow conditions, boundary conditions, and the structural parameters of the cylinder.

Figure 5(top) illustrates some of the uncertainties in the cylinder flow. This study employs the generalized polynomial chaos [6] to solve such stochastic flow problems. Figure 5(bottom) shows the spatial-temporal contours of the lift force for a deterministic flow (top plot) and a stochastic flow (bottom three plots) past a cylinder at the Reynolds number Re = 300. The stochastic flow solution corresponds to an uncertainty of 1% in the incoming

4b

4a

Figure 4. Deterministic flow past a rigid cylinder oscillating freely in cross-flow direction at Re=1000. (a) Spatial-temporal distributions of the drag (top) and lift (bottom) forces on the cylinder. (b) pressure isocontour in cylinder wake.

flow. The plots show the contributions to the lift force from different random modes. The contribution from the 0th (mean) random mode decreases, while the contributions from the higher random modes grow over time.
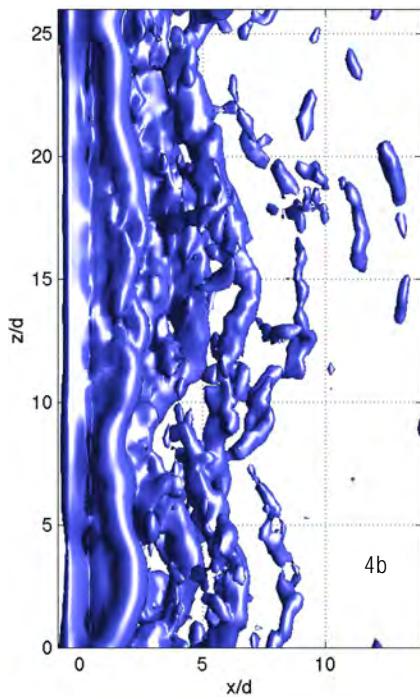
SUMMARY

Parallel computations exploiting a single-level parallelism for VIV simulations have clear performance limitations that preclude scaling to the large number of processors on modern supercomputers. To take advantage of the hierarchical structures inherent in VIV computations, this article has presented two multilevel parallel paradigms based on MPI/MPI and MPI/OpenMP in the context of spectral element methods that completely eliminate the performance restrictions in single-level parallel computations. Because a greatly reduced number of processes are involved in the communications at each level, these multilevel parallel paradigms reduce the network latency overhead and enable the applications to scale to a large number of processors more efficiently. The multilevel parallel paradigms presented here are suitable for VIV computations at high Reynolds numbers.
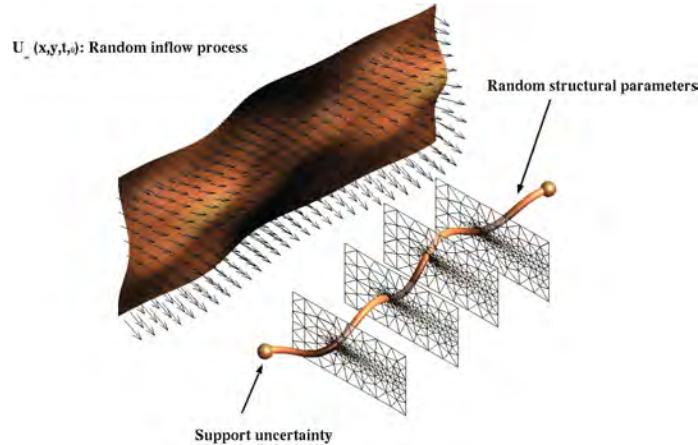
Figure 5. Stochastic flow past a cylinder. Top: schematic of a random inflow process past a cylinder with random boundary conditions and random structural parameters; Bottom: Spatial-temporal contours of the lift force for a deterministic flow (a) and a stochastic flow (b-d) past a cylinder at Re=300; the bottom three figures show the lift force contributions by the 0th random mode (b), 1st random mode (c), and 2nd random mode (d).
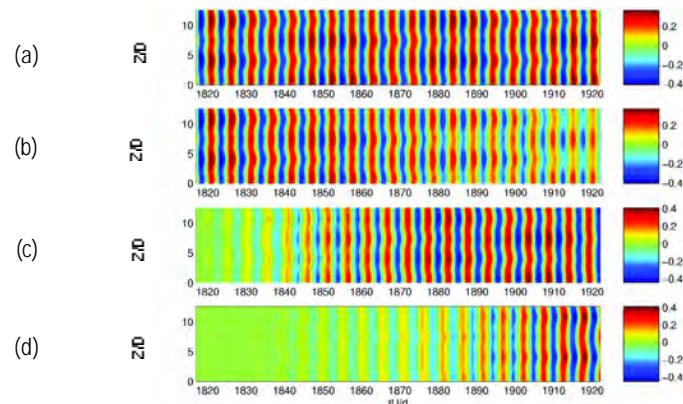
References

1. Evangelinos, C. and G.E. Karniadakis, "Dynamics and Flow Structures in the Turbulent Wake of Rigid and Flexible Cylinders Subject to Vortex-Induced Vibrations," Journal of Fluid Mechanics, 400, 91, 1999.

2. Newman, D. and G.E. Karniadakis, "A Direct Numerical Simulation Study of Flow Past a Freely Vibrating Cable," Journal of Fluid Mechanics, 344, 95, 1997.

3. Dong, S. and G.E. Karniadakis. "Dual-level parallelism for deterministic and stochastic CFD problems," Proceedings of Supercomputing 2002, Baltimore, Nov. 2002.

4. Dong, S. and G.E. Karniadakis. "P-refinement and P-threads," Computer Methods in Applied Mechanical Engineering, 192, 2191-2201, 2003.

5. Gopalkrishnan, R., "Vortex-Induced Forces on Oscillating Bluff Cylinders," Ph.D. Dissertation, Department of Ocean Engineering, Massachusetts Institute of Technology, 1993.

6. Xiu, D., D. Lucor, C.H. Su, and G.E. Karniadakis, "Stochastic Modeling of Flow-Structure Interactions Using Generalized Polynomial Chaos," Journal of Fluid Mechanics, 124, 51-59, 2002.

SALMON Virtual Environment: A Collaboration Between the NAVO MSRC and ARSC

Pete Gruzinskas and John Van der Zwaag, NAVO MSRC Visualization Center

The NAVO MSRC provides tailored support to the Climate, Weather, and Ocean (CWO) modeling community. This includes support in the realm of High Performance Computing (HPC), storage, and visual analysis. In the area of visual analysis, the NAVO MSRC Visualization Center has become proficient in developing interactive analysis applications, which allow users to interactively analyze and interrogate their full-resolution model output. These custom applications are efficient and portable and can usually be run on the users' desktops.

In recent history, the NAVO MSRC Visualization Center has collaborated with users, other Shared Resource Centers (SRCs), and other Government agencies to provide no-cost, custom, portable analysis routines.

In a continuing effort to provide users (in this case Dr. Kate Hedstrom and the Sea-Air-Land Modeling and Observing Network (SALMON)) with visual analysis solutions, a collaboration with the Arctic Region Supercomputing Center (ARSC) was initiated to exploit the ARSC's new state-of-the-art immersive display system (CAVE). Testing at the Mississippi State University (MSU) CAVE concluded this software application project, which should be delivered by November 2003.

DESCRIPTION

The Cave Automatic Virtual Environment (CAVE), originally developed at the University of Illinois, is a room-sized, multi-user virtual reality device. A CAVE consists of one to six walls on which Three-Dimensional (3D) stereo graphics are projected and usually includes some type of spatially tracked input device such as a wand or pinch glove. By tracking the position and orientation of the user's head, the graphics can be generated in such a way to give the user a very immersive experience. The CAVE has been useful in a variety of scientific visualization applications such as vehicle design and ocean simulation.

While 3D immersion, as provided by CAVE, is not necessarily the panacea of analysis, in the case of CWO it allows analysts to literally enter the

Right: User in CAVE environment.

Below: CAVE GUI interface with particle trace.
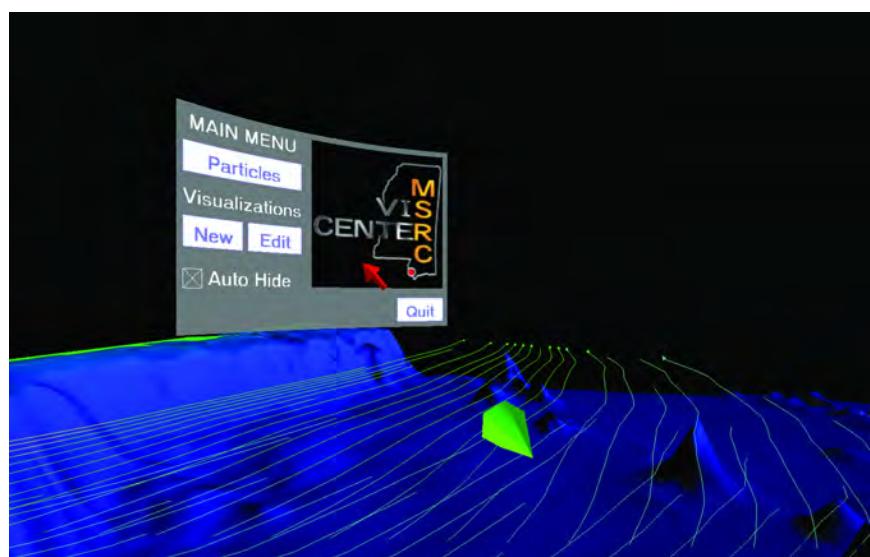
environment and view 3D ocean features from the inside out, unlike any other display device.

HARDWARE ISSUES

One of the biggest challenges was the significant geographic separation between the development site and the target hardware environment. Consequently, CAVE was developed and tested without the benefit of using the actual production hardware. This challenge was overcome through the use of VRJuggler and collaboration with MSU.

A key decision was to use the VRJuggler libraries. VRJuggler, developed at the Iowa State University Virtual Reality Center, is a suite of tools used for developing virtual environments without needing to understand the underlying hardware. Applications developed in VRJuggler can easily be reconfigured to run on anything from a CAVE system to an Immersadesk. VRJuggler is also cross-platform and can be run using a simulator mode so development can be accomplished locally on a Linux workstation. Another benefit of VRJuggler is that it is an open source project and does not require licensing fees to use.

While initial testing could have been performed locally on a desktop workstation, issues such as multi-threading, multiple displays, and input devices required that final testing be performed on an actual CAVE system. Even though travel to ARSC was certainly a possibility, it was more beneficial, both in regards to cost and time, to use a similar system at a closer location. This provided the perfect opportunity to collaborate with MSU, whose recently upgraded computerized virtual environment (COVE) system closely matches the CAVE system and is conveniently located within driving distance of the NAVO MSRC.

DESIGN ISSUES

One of the most important issues in designing a virtual environment in a system like a CAVE is interactivity. In a CAVE system, the user's head is tracked so the correct projection can be displayed on the walls according to the user's position in the physical environment. If the frame rate is not sufficiently high, it can destroy the immersive feeling and even cause disorientation or nausea.

One of the easiest ways to ensure interactive frame rates is to take advantage of the multiple processors available on SGI Onyx systems. A VRJuggler application is designed to be separated across multiple threads by dedicating a separate thread for
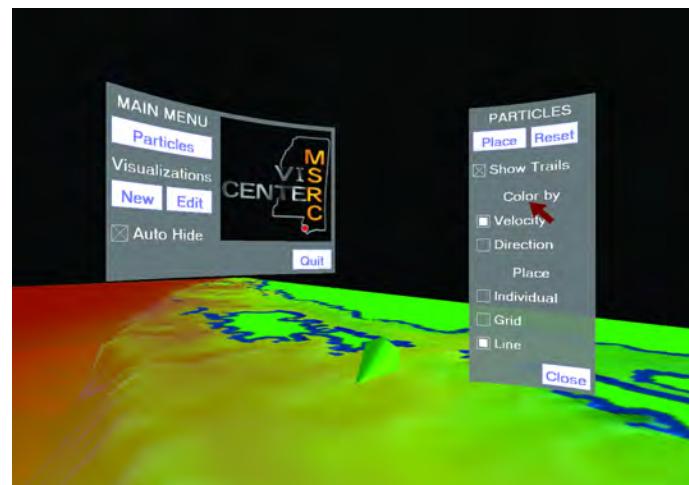
CAVE GUI interface with a horizontal slice view.
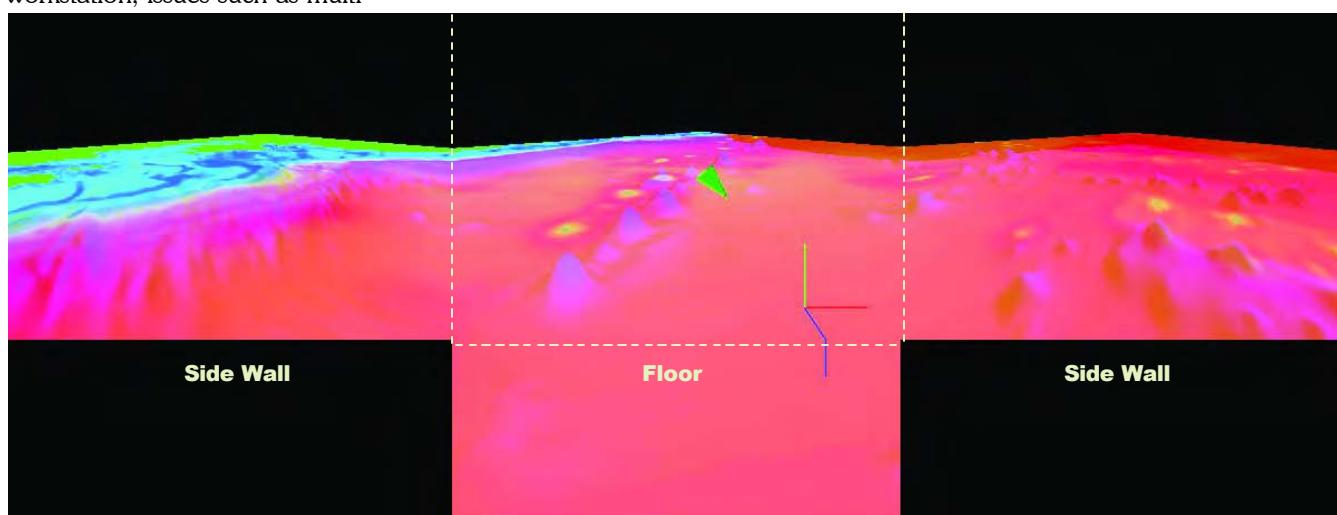
Flattened 3D CAVE environment which simulates what would be rendered on the individual walls.

Performance Analysis Tools Newly Deployed at NAVO MSRC

Shirley Moore, Innovative Computing Laboratory/University of Tennessee - Knoxville
PET Computational Environments Functional Area Point of Contact

If you are involved in developing and/or maintaining parallel application code, you are no doubt concerned about understanding and improving the code's performance. Two technologies that can help with these tasks are the Performance Application Programming Interface (PAPI) cross-platform interface to hardware performance counters and the Tuning and Analysis Utilities (TAU) suite of performance analysis tools. Together, these technologies allow the user to easily collect accurate and relevant performance data and analyze the data in terms of application source code constructs.

PAPI

PAPI provides a portable interface to the hardware performance counters available on most modern microprocessors. These counters exist as a small set of registers that count events, which are occurrences of specific signals related to the processor's functions. Monitoring these events helps in understanding the correlation between the structure of source/object code and the efficiency of the mapping of that code onto the underlying architecture. This correlation has a number of uses in performance analysis, including performance modeling, benchmarking, hand tuning, and effective use of compiler optimizations.

PAPI provides two interfaces to the underlying counter hardware; a simple, high-level interface for the acquisition of simple measurements and a fully programmable, low-level interface directed toward tool developers and users with more sophisticated needs. The low-level interface is fully programmable and has features such as guaranteed thread safety and writing of counter values, multiplexing, and notification on threshold crossing as well as processor-specific features. The high-level interface simply provides the ability to start, stop, and read specific events one at a time. PAPI provides portability across different platforms by using the same routines with similar argument lists to control and access the counters for every architecture.

As part of PAPI, a predefined set of events has been established to represent the lowest common denominator of every good counter implementation. The intent is that the same source code will count similar

```
TAU_PROFILE_TIMER(t1,"multiply-regular", "void (v
oid)", TAU_USER); []
 for (n = 0; n < SIZE; n++)
   for (m = 0; m < SIZE; m++)
   {
     A[n][m] = B[n][m] = n + m ;
     C[n][m] = 0;
   }
 TAU_PROFILE_START(t1);
 for (i = 0; i < SIZE; i ++)
   for (j = 0; j < SIZE; j++)
     for (k = 0; k < SIZE; k++)
       C[i][j] += A[i][k] * B[k][j];
 TAU_PROFILE_STOP(t1); /* FIG A: ORIGINAL */
```

```
TAU_PROFILE_TIMER(t1,"multiply-regular", "void (void
)", TAU_USER); []
 for (n = 0; n < SIZE; n++)
   for (m = 0; m < SIZE; m++)
   {
     A[n][m] = B[n][m] = n + m ;
     C[n][m] = 0;
   }
 TAU_PROFILE_START(t1);
   for (i = 0; i < SIZE; i ++)
     for (k = 0; k < SIZE; k++)   /* ijk->ikj */
       for (j = 0; j < SIZE; j++)
         C[i][j] += A[i][k] * B[k][j];
 TAU_PROFILE_STOP(t1); /* FIG B: LOOP INTERCHANGE */
```

```
TAU_PROFILE_START(strip_timer);[]
for(i=0; i < SIZE; i++)
   for(sz = 0; sz < SIZE; sz+=CACHE)
   {
     //vl = min(SIZE-sz, CACHE):
     vl = (SIZE - sz < CACHE ? SIZE - sz : CACHE);

     for(strip = sz; strip < sz+vl; strip++)
       for(k=0; k < SIZE; k++)
         C[i][strip] += A[i][k]*B[k][strip];
   }
TAU_PROFILE_STOP(strip_timer);
/* FIG C: STRIP MINING OPTIMIZATION */
```

```
TAU_PROFILE_START(strip_timer);[]
for(i=0; i < SIZE; i++)
   for(sz = 0; sz < SIZE; sz+=CACHE)
   {
     //vl = min(SIZE-sz, CACHE):
     vl = (SIZE - sz < CACHE ? SIZE - sz : CACHE);
       for(k=0; k < SIZE; k++)
         for(strip = sz; strip < sz+vl; strip++)
           C[i][strip] += A[i][k]*B[k][strip];
   }
TAU_PROFILE_STOP(strip_timer);
/* FIG D: LOOP INTERCHANGE WITH STRIP MINING */
```

Figure 1. Four variants of the matrix multiply algorithm.

and possibly comparable events when run on different platforms. If the programmer chooses to use this set of standardized events, then the source code need not be changed, and only a fresh compilation and link is necessary. However, should the developer wish to access machine specific events, the low-level Application Programming Interface (API) provides access to all available events and counting modes.

TAU

TAU is an integrated toolkit for performance instrumentation, measurement, and analysis of parallel message passing and/or multi-threaded programs. TAU provides a portable profiling and tracing toolkit for the performance analysis of applications written in C++, C, Fortran 90, Java, and Python.

While profiling computes summary statistics of performance metrics (such as the total exclusive and inclusive time/counts spent in a routine), tracing captures detailed time-stamped event logs that highlight the time-varying aspect of performance events. TAU uses PAPI to access low-overhead wallclock time and hardware performance counters and to process virtual time. TAU supports different modes of instrumentation: source code, pre-processor-based rewriting of source

Figure 2. TAU report of program executed on a Pentium III/500 MHz Xeon processor with PAPI.

code, Message Passing Interface (MPI) library level, and runtime instrumentation. The instrumentation captures data for functions, methods, basic blocks, and statement execution at all execution levels. An API provides selection of measurement groups for organizing and controlling instrumentation.

A pre-processor-based instrumentor (tau_instrumentor) rewrites the original code with TAU instrumentation calls. It uses Program Database Toolkit (PDT) to parse the application (written in F90, C/C++) and subsequently traverse the abstract syntax tree to insert calls to start and stop TAU timers at routine transitions. TAU provides both command-line and

graphical tools for analyzing profile data. Trace files produced by TAU instrumentation can be converted into a number of different formats for viewing by third-party trace analysis tools, such as VAMPIR.

EXAMPLE

To illustrate the use of TAU and PAPI, consider four variants of the well-known matrix multiply algorithm as shown in Figure 1. In this example, a timer with a name, signature, and group parameters is declared. The timer start and stop calls around the main loop are used to measure a variety of performance metrics. The TAU has been configured using:

```
% configure -papi=<dir> \
MULTIPLECOUNTERS \
PAPIWALLCLOCK
```

This creates a TAU stub makefile that is included in the application makefile for compiling and linking the TAU and PAPI libraries. (Note that on NAVO MSRC machines, applying configuration is unnecessary, since TAU has already been installed as part of the computational environment—see the note at the end of this article.)

The following environment variables were set before executing the program:
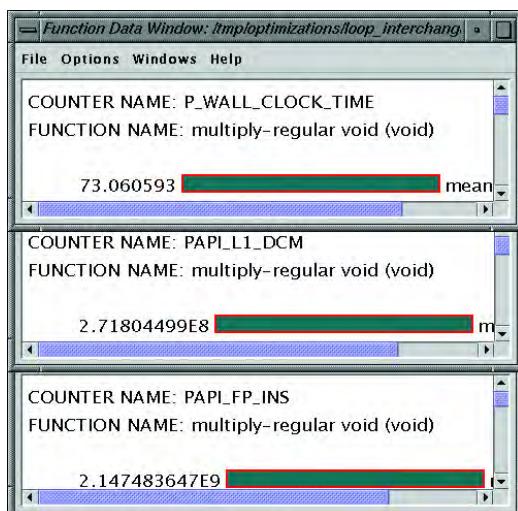
Figure 3. Loop interchange optimization employed where the *ijk* loop is transformed to the *ikj* loop.
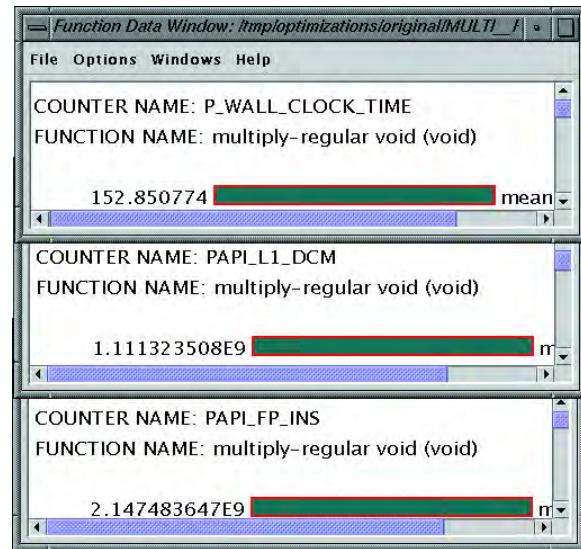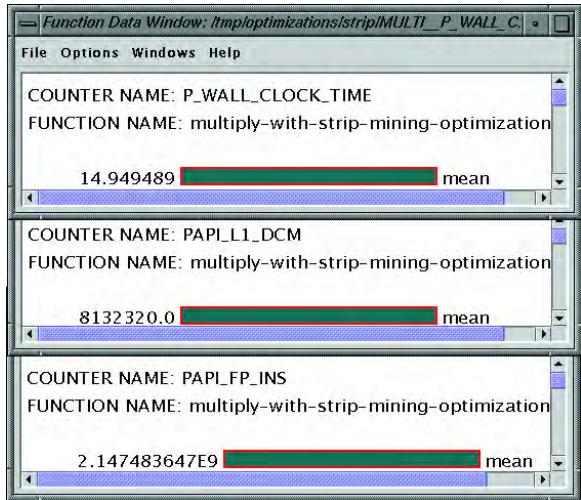
Figure 4. Strip mining optimization applied to the original code.

```
% setenv COUNTER1 \
PAPI_FP_INS

% setenv COUNTER2 \
PAPI_L1_DCM

% setenv COUNTER3 \
P_WALL_CLOCK_TIME
```

This allows measurement of the number of floating point instructions executed, level 1 data cache misses, and the wallclock time, respectively.

When the program is executed on a Pentium III/500 MHz Xeon processor, it takes 152.85 seconds of wallclock time, executes 2.14E9 floating point operations, and incurs 1.11E9 data cache misses in the main instrumented loop as reported by TAU using PAPI (see Figure 2). This verifies that the number of floating point instructions executed matches the expected matrix multiply loop execution (for problem size n=1024, $2*n^3 = 2.14E9$).

Figures 3, 4, and 5 show how the loop that computes the product of the two matrices can be rewritten. In Figure 3, the loop interchange optimization is employed where the *ijk* loop is transformed to the *ikj* loop. This reduces the number of data cache misses and consequently the time to execute the loop (to 73 seconds).

In Figure 4, to further reduce the data cache misses, the strip mining optimization is applied to the original code by performing the computation on strips of size 128 (CACHE in the code). This reduces the execution time to 14.9 seconds.

Finally, in Figure 5, the strip-mining and loop-interchange optimizations are combined. This reduces the number of data-cache misses from 1E9 to 7.1E5 and the wallclock time from 152 seconds to 11.2 seconds.

CONCLUSIONS

Parallel performance problem solving depends on robust systems for empirical performance evaluation. Flexibility and portability in empirical methods and processes are influenced primarily by the strategies available for instrumentation and measurement and how effectively they are integrated and composed.

The matrix multiply example demonstrates how performance data from hardware performance counters is critical in identifying causes for poor performance.

PAPI and TAU have been installed and are ready for use on the NAVO IBM POWER3 (HABU) and POWER4 (MARCELLUS) platforms. For more information, see the PAPI and TAU documentation in the Computational Environments (CE) section of the Online Knowledge Center (OKC) (from the top level OKC page, scroll down on the left and click on CE, then scroll down on the CE main page to FY03 Projects and click on "Consistent Well-documented Computational Environment," where you will find links for PAPI and TAU), or contact CE at pet-ce@cs.utk.edu. For information specifically about where PAPI and TAU are installed on NAVO MSRC machines, see the PET Computational Environments repository at http://rib.cs.utk.edu/cgi-bin/catalog.pl?rh=355.
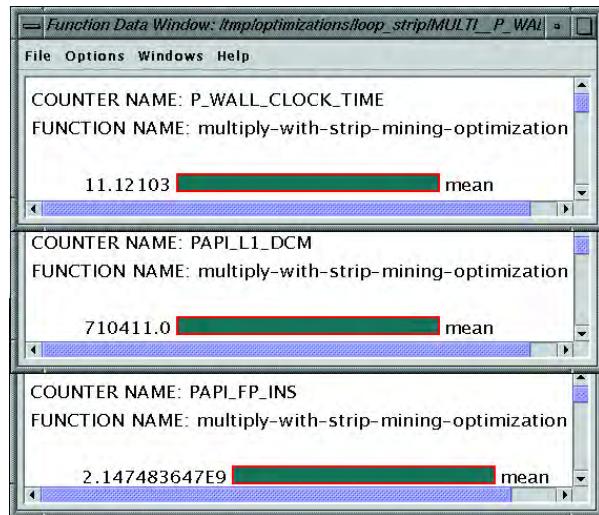
Figure 5. Combined strip-mining and loop-interchange optimizations.

Density Functional Theory Studies of GaN Surface Reactivity

A. A. Gokhale, J. D. Schieke, M. Mavrikakis, T. F. Kuech
Department of Chemical and Biological Engineering
University of Wisconsin - Madison

With the increasing application of direct band gap materials like Gallium Nitrogen (GaN) in short wavelength optoelectronics and high temperature, high power electronics, greater focus has shifted to reducing the number of defects in these materials, so that better control can be achieved over the material properties. By altering the surface energetics, surfactants like Antimony (Sb) influence a variety of processes such as adsorption, desorption, nucleation, surface diffusion, and step edge growth of GaN. As a result, such surfactants are used during semiconductor film growth to improve the control over the processes that govern the growth of semiconductor materials. Among other factors, crystallographic orientation plays a significant role in the growth process. More specifically, ab initio studies are intended to improve our understanding of the surfactant effects on different GaN facets resulting into Lateral Epitaxy Overgrowth (LEO).

A planewave total energy calculation code, DACAPO, which implements periodic boundary conditions, is used. The Kohn-Sham one-electron valence states are expanded in a basis of plane-waves with kinetic energies below 25Ry, and the Generalized Gradient Approximation (GGA-PW91) is used to describe the exchange-correlation energy. Ultrasoft pseudopotentials are employed to describe the ionic cores. A slab consisting of eight layers of GaN($000\bar{1}$) or GaN($11\bar{2}0$) and accounting for the appropriate dipole correction is used to model these surfaces. In all cases, the top four GaN layers are allowed to relax to their equilibrium positions, while the bottom four layers are constrained to their ideal bulk positions. The Brillouin zone is sampled with a 4x4x1 Monkhorst-Pack grid. The self-consistent PW91 density is determined by iterative diagonalization of the Kohn-Sham states (kb=0.1eV) and Pulay

This figure illustrates the electron density change upon adsorption of Antimony Nitride (SbN) on a GaN(0001) surface. First principles, periodic, and Density Functional Theoretical calculations performed by researchers at the Department of Chemical and Biological Engineering, University of Wisconsin - Madison, using NAVO MSRC computational resources, suggest that SbN formation plays an important role in the surfactant action of Sb in GaN's growth. The magnified inset shows the changes in the electron density projected onto a plane perpendicular to the GaN(0001) surface and passing through the Sb and N atoms, occurring when SbN adsorbs on the GaN(0001) surface. Shades of brown and red indicate a progressive increase in the electron density, while shades of green and blue signify a progressive decrease in the electron density, respectively.

Figure 1. Schematic of adsorption on GaN(0001) of: (a) Sb at fcc site, (b) Sb at hcp site, (c) Ga at fcc site, (d) N at hcp site.

mixing of the resulting electronic density. Convergence with respect to slab thickness, k-point set, and cut-off energy is always confirmed.

Our calculations show that bulk incorporation of Sb in GaN is highly unfavorable, suggesting that Sb tends to segregate on the surface. On the GaN(0001) surface, Sb can bind at either the fcc or hcp sites with almost equal strengths, as shown in Figure 1a and 1b, respectively. This is demonstrated by the identical binding characteristics of the Sb atom at both the Face-Centered Cubic (fcc) and Hexagonal Close Packing (hcp) sites,

as shown by the electron density plots in Figure 2. For comparison, Ga shows a clear preference for the fcc site, whereas N shows a clear preference for the hcp site on the GaN(0001) surface, as shown in Figure 1c and 1d, respectively.

Once the thermochemistry of atomic and molecular adsorption has been studied, the kinetics of the individual elementary steps (e.g., diffusion of the surface, bond-breaking/making events, etc.) is investigated using the Nudged Elastic Band (NEB) method. The final goal of these costly iterative calculations is to determine the

minimum energy path and the corresponding activation energy barrier. The stable endpoints of all NEB calculations are allowed to fully relax to their respective local minima. These endpoints are connected through a set of 7 to 10 intermediate images on the elastic band. The intermediate images are subsequently allowed to relax in all degrees of freedom except for the direction of the band itself, until the energy landscape converges to the minimum energy path connecting the two endpoints. A cubic spline energy interpolation scheme between the

Figure 2. Charge density difference plot of Sb adsorbed on the fcc and hcp sites of a GaN(0001) surface. The plots are obtained by subtracting slab charge density plus gas phase atomic charge density from the composite system's charge density. Contours are spaced 0.01 eV/Å3 apart, with the 0.0 eV/Å3 contour in black. Yellow and red indicate electron density decrease, while blue and green indicate electron density increase.
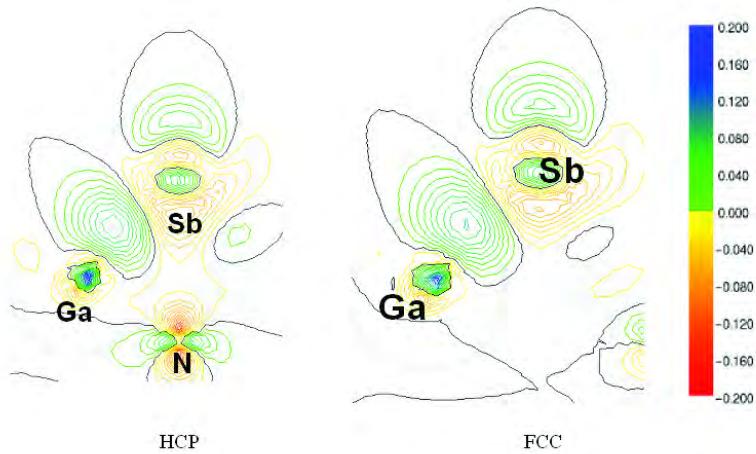
images is implemented, using the calculated Hellman-Feynman forces along the reaction path, in order to obtain an accurate estimate for the true transition state. Transition states identified with the NEB method are verified by vibrational frequency analysis yielding a single imaginary frequency.

The NEB calculations performed in this study suggest that atomic nitrogen (N) has a diffusion barrier of 1.0 eV on GaN(0001) surface, whereas Sb has a diffusion barrier of ca. 0.5 eV on the same surface. The higher mobility of Sb is commensurate with its surfactant action.

Furthermore, the recombination reaction N(a) + N(a) à N2 (g) on the GaN(0001) surface, is rather difficult with an activation barrier of 1.9 eV, with the reverse reaction being even more difficult (activation barrier = 3.1 eV). As an alternative to this recombination reaction of N(a), the authors found the barrier to the formation of SbN(a) to be far easier (0.7eV), which makes this a feasible reaction path under most experimental growth conditions.

Once formed, SbN(a) diffuses across the surface, with a barrier of 0.7 eV, increasing the effective mobility of the N atom on the GaN(0001) surface until the next surface event occurs. This can be either the incorporation of an N atom at a step edge (Figure 3a) or the recombination with another N atom, in which case N2 is formed and desorbed from the surface (Figure 3b). In both cases, the N atom is released from SbN(a), freeing up the Sb(a) to react with another N(a) atom and repeat the cycle.

This cycle continuously regenerates Sb(a), and hence only a small amount of surface Sb would be needed for the above catalytic cycle to operate. This is in accord with the fact that under typical growth conditions, the vapor pressure of Sb is several orders of magnitude higher than that of gallium, suggesting that only a small amount of Sb will be on GaN(0001) surface available for surfactant action.

The studies described in this article show that the surfactant action of Sb is present on the GaN(1120) surface too, but the kinetics of the elementary steps are quite difficult there. This leads to a remarkable increase in the lateral overgrowth rate and results in formation of vertical facets on GaN in the presence of even low partial pressures of Sb.

Because of the deep surface relaxations characterizing the GaN surfaces, a large number of GaN layers need to be included in the total energy calculations. Accordingly, massively parallel computing environments, such as those available at the NAVO MSRC, make an ideal platform for pursuing this fundamental research.
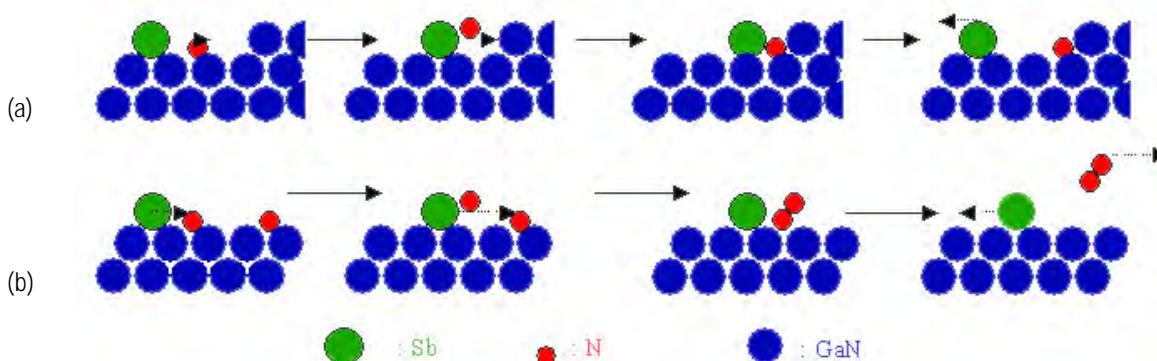
Figure 3. Schematic representation of Sb interaction with N, and N transport on GaN(0001) surface. (a) SbN formation, followed by N deposition at a GaN step-edge/defect. (b) SbN formation, followed by N2 formation and desorption of N2 from the surface.

References

1. Zhang, H. F. Tang, J. Schieke, M. Mavrikakis, and T.F. Kuech, "The addition of Sb as a Surfactant to GaN Growth by Metal Organic Vapor Phase Epitaxy,"Journal of Applied Physics, 92, 2304 (2002).

2. Zhang, L., H. F. Tang, J. Schieke, M. Mavrikakis, and T.F. Kuech, "Influence of Bi Impurity as a Surfactant During the Growth of GaN by Metalorganic Vapor Phase Epitaxy,"Journal of Crystal Growth, 242, 302 (2002).

THE PORTHOLE
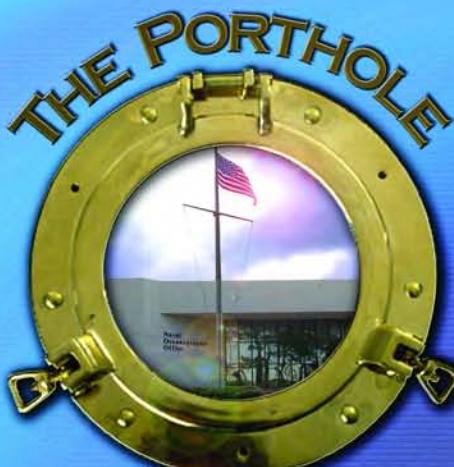
1. NAVOCEANO Naval personnel in formation during the annual September 11th Remembrance Ceremony.
2. Visit of CDR Antonio Claudio M. Vieira, Brazilian Navy Hydrographic Center; Kathryn Townsend, NAVOCEANO; Dave Cole, NAVO MSRC.
3. New Meteorology and Oceanography students.
4. Steve Adamec, NAVO MSRC Director; Dave Wennergren, DON CIO; Bill Richard, EDS; Tom Dunn, CNMOC CIO.
5. New members of the NAVOCEANO Fleet Survey Team.
6. Melanie Gehman, NAVOCEANO Public Affairs Office; Dave Cole, NAVO MSRC; Mr. Marty Morgan, Historian at the D-Day Museum in New Orleans; Steve Faber, NAVOCEANO.
7. Lockheed Martin Space Operations VIP tour. Steve Adamec, NAVO MSRC Director; Ken Reightler, LM; Jay Honeycutt, LM; Linda Wise-Pyfrom, LM; Bobby Knesel, NAVO MSRC Deputy Director; Cleon Lacefield, LM; Don Fulop, LM.
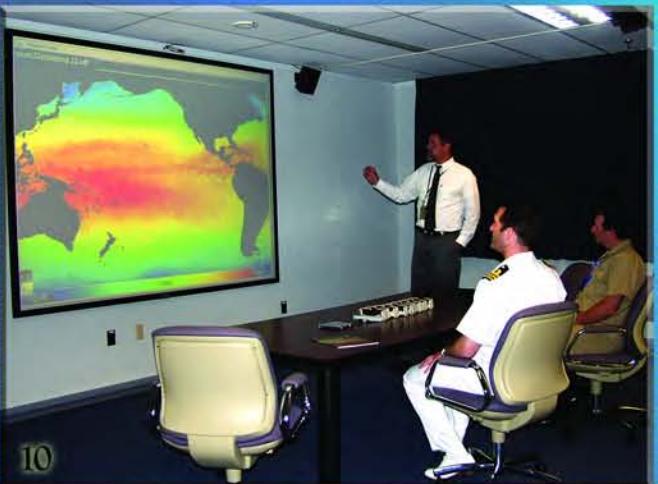
8. NAVO MSRC hosted Aeronautical Systems Center (ASC) MSRC Visualization team members, (front) Rhonda Vickery and Brian Schafer. Back row: John van der Zwaag, LMSO; Peter Flynn, NRL; Tim Campbell, PET; Chad Saxon, LMSO; Brian Tabor, PET; Pete Gruzinskas, NAVO MSRC.

9. LM Corporate Virtual Environment Technology Focus Group.

10. Pete Gruzinskas demonstrates Global Sea Surface Temperature generated from NLOM Ocean Circulation Model to CDR Laurence Smallman, Royal Navy, and LCDR Don Ventura, Royal Navy.

11. NIMA Mission Specific Data Set Team.

12. Middle School teachers participate in the NASA Summer Education Program.

13. Partners for Stennis Test Pilot for Schools view a presentation in the NAVO MSRC Visualization Center.

NAVO MSRC PET Update

Eleanor Schroeder, NAVO MSRC Programming Environment and Training Program (PET) Government Lead

As we continue into the third year of the revamped PET program, we are beginning our third year of project efforts. The time duration of the projects will be a bit shorter this year—eight months—since we are trying to get that timeline aligned with that of the contract year. In June 2004, PET will begin to exercise the option years.

For Component 1, we have several projects in the works.

CWO-04-001: INFRASTRUCTURE DEVELOPMENT FOR REGIONAL COUPLED MODELING ENVIRONMENTS

Software infrastructure developed in PET to support coupled simulation of complex geophysical phenomena will be applied to full-up, large-scale coupled environments and ecosystem models, such as High Fidelity Simulations of Littoral Environments (HFSoLE), in combination with the Weather Research Framework (WRF) model.

EQM-04-002: ERROR ESTIMATORS/INDICATORS FOR ENVIRONMENTAL QUALITY MODELING

This effort will investigate error estimators for coupled flow and reactive transport problems and study compatibility of algorithms for flow and transport to determine the minimal requirements on flow algorithms needed to maintain accuracy and conservation properties of the numerical method used for transport.

EQM-04-003: LINEAR AND NONLINEAR SOLVERS FOR ENVIRONMENTAL MODELING

This effort will extend ideas developed in the context of finite difference methods on structured grids in the parallel simulator Integrated Parallel Accurate Reservoir Simulator (IPARS) at the University of Texas at Austin to unstructured grids and other types of discretizations, including finite element methods based on continuous and/or discontinuous approximating spaces.

CE-04-001: A CONSISTENT WELL-DOCUMENTED COMPUTATIONAL ENVIRONMENT

This ongoing effort is building a consistent, well-documented computational environment across the HPC centers: compilers, message passing libraries, numerical libraries, debugging and performance analysis tools, and data management and visualization tools.

CE-04-002: PAPI DEPLOYMENT, EVALUATION, AND EXTENSIONS

In this ongoing effort, PAPI, a cross-platform interface to hardware performance counters, is being deployed and supported on all HPC center platforms and kept up to date with processor and operating system upgrades.

CE-04-003: APPLICATION PORTABILITY

This effort includes user outreach for a PET-developed portability knowledge base that tracks bugs, nonconforming features, and common programming mistakes, as well as continued maintenance and transition support of the knowledgebase to HPC center staff.

CE-04-007: ENHANCING DoD HPC RESEARCH THROUGH CLUSTERS

This effort will develop a PET Cluster User's Guide and a PET Cluster Administrator's Guide to provide instructions for building robust clusters using National Partnership for Advanced Computational Infrastructure (NPACI) Rocks.

CE-04-008: SCALABILITY & PERFORMANCE OPTIMIZATION TEAM (SPOT)

This project will build on the "Consistent and Well-Documented Computing Environment" PET project to optimize strategic DoD HPC applications and also to produce detailed case studies on how these codes were profiled, how the tools were used, and what results were accomplished.

More information about these projects and projects funded in previous years can be found on the PET Online Knowledge Center: https://okc.erdc.hpc.mil. If you haven't visited it lately—or if you haven't already established an account on it—you really should. There's a wealth of information available there.

In August 2003, we closed out the summer intern program. We were fortunate to have two students based here at the NAVO MSRC, Derrick Johnson from Georgia Institute of Technology and Xavier University in Atlanta, Georgia and Angela McClure from Central State University in Wilberforce, Ohio. Both students participated in the PET Summer Intern Presentations that were broadcast across the Access Grid. An article about what Derrick and Angela did during the summer follows.

PET Summer Interns Focus on MPI and OpenMP Super Computing Environments

This summer we were pleased to have two enthusiastic interns working with Dr. Tom Cortese, the PET Computational Environments onsite: Derrick Johnson and Angela McClure. Mr. Johnson is starting his senior year at the Georgia Institute of Technology (Georgia Tech.) this fall and is studying Computer Engineering and Physics. Ms. McClure is a graduating senior from Central State University in Wilberforce, Ohio, with dual degree in Computer Science and Mathematics and plans to pursue a Ph.D. in Bioinformatics. Both focused on aspects of parallel computing, especially the use of Message Passing Interface (MPI) and OpenMP to facilitate better programming and communication. The following is compiled from their final reports.

DERRICK JOHNSON - PARALLEL COMPUTING WITH MPI AND PERFORMANCE ANALYSIS

Why research parallel computing? Parallel computing provides the computational power for everything from predicting the path of hurricanes to producing blockbuster animated movies such as "The Hulk." This ability to produce faster, cheaper, and more efficient systems has always been the goal of scientists and engineers.

This summer, as part of my internship with the PET program, I undertook the study of the performance of several computational models to possibly find regions of poor scaling or other regions where performance could be improved. But, before this could be accomplished, background information in the Unix operating system and shell scripting, multiprocessor architecture, and parallel programming with MPI and OpenMP were needed as foundation material.

Since all of the computers that were utilized during the internship were Unix systems and most of the computers at universities use the Windows® operating system, there was a slight learning curve. Through trial and error it was learned that the best way of submitting multiple jobs was not through manual labor, rather by automating this task through shell scripting. By the end of the summer, indispensable knowledge was gained about the Unix operating system, and several scripts were created to reduce some of the toil involved with submitting multiple

jobs requesting varying number of processors.

In addition, knowledge of multiprocessor architecture was needed in order to understand how to better coordinate parallel processors and how processors share data. Processors with a single address space, sometimes called shared-memory processors, offer the programmer a single memory address space that all processors share. Processors communicate through shared variables in memory, with all processors capable of accessing any memory location via loads and stores.

Single-address-space multiprocessors come in two styles. Uniform Memory Access (UMA) multiprocessors take the same time to access main memory no matter which processor requests it and no matter which word is asked. In a Nonuniform Memory Access (NUMA) machine some memory accesses are faster than others, depending on which processors are involved in the data transfer.

As might be expected, there are more programming challenges to get the highest performance from a NUMA multiprocessor than a UMA multiprocessor, but NUMA machines can scale to larger sizes and hence offer potentially higher performance. The alternative model to shared memory for communicating uses message passing for communicating among processors. Message passing is required for machines with private memories, as opposed to shared memory.

It is difficult to write code in parallel because of the overhead of the communication. As an example, think of the burden of communication for a task done by two people compared to the burden of a task done by a group, especially as the size of the committee increases. Another reason why it is difficult to write parallel-processing programs is that the programmer must know a good deal about the hardware. On a one-processor machine, the programmer writes the program, largely ignorant of the underlying machine organization; issues such as data layout are handled mainly by the compiler. In order to write a program that takes full advantage of the underlying hardware, the programmer must know some information about the topology of the machine. And this is a trade off-tailoring a program in this manner will hinder its ability to be ported to another machine.

In order to better understand these difficulties and learn how to write parallel code, I studied parallel programming with MPI. Starting from a serial code, I learned how to

completely restructure the data layout in order to produce a parallel version, using MPI, which I was then able to run on several multiprocessor architectures. The efficiency of this code was tested by plotting several graphs to ensure that the time required for program execution was decreasing as more processors were added to the job. Vampir and VampirTrace from Pallas, performance analyze tools, were used to analysis the data and to ensure that the processors were communicating properly.

Finally, Simulating WAves Near shore (SWAN), an important third-generation wave model used to simulate short-crested, wind-generated waves in shallow water areas such as coastal regions and inland waters, was chosen to be the subject of the performance analysis.

In order to understand the parallel performance of SWAN it is helpful to explore two quantities known as speedup and efficiency. Speedup is the time it takes for one processor to finish a task over the time it takes for N processors to finish the same task. Efficiency is speedup over N processors. Theoretically, the speedup can never exceed the number of processors. The efficiency is a measurement of the fraction of time for which a processor is usefully employed. In an ideal parallel system the speedup is equal to N, and the efficiency is equal to one (i.e., doubling the number of processors reduces the execution time by a factor of two). In practice the speedup is less than N, and efficiency is between zero and one, depending on the design of the parallel system and the parallel program. The results of this performance analysis showed that the performance of the MPI and OpenMP version of the SWAN code as closely related in efficiency.

This summer internship has truly been both challenging and rewarding. We had opportunities to learn about everything from how the cache works to programming with MPI. This experience will also help me in my future endeavors by providing background information in parallel programming and MATLAB, two tools that are requirements for my upcoming classes for the fall semester. Overall, I have enjoyed my time here and the time my mentor (Tom Cortese) and colleagues have taken out of their busy schedules to sit down to lecture and talk to us about anything we needed help with. I would highly recommend working at the NAVO MSRC to anybody who would like to learn about High Performance Computing.

ANGELA MCCLURE - DEBUGGING AND PERFORMANCE ANALYSIS

The Department of Defense (DoD) relies on High-Performance Computing (HPC) as a key enabling technology. Parallel programming skills, including debugging and performance analysis techniques, are

powerful and marketable tools in today's HPC environment. My goals in participating in the internship program were to acquire knowledge of MPI and OpenMP and to gain a better understanding of how to analyze MPI and OpenMP programs with debugging and performance analysis tools.

Under the tutelage of Tom Cortese, the Computational Environment on-site, I learned about the fundamentals of parallel programming, including using MPI and OpenMP, as well as became familiar with debugging tools and benchmarking.

The Stommel Model is a two-dimensional solution to a model problem for Ocean Circulation that provides a good basis for learning various parallel programming skills. Using this model, I learned how the data were organized so the model (originally written in serial Fortran, now parallelized with both MPI and OpenMP) could be run utilizing parallel processing machines. TotalView, an interactive graphical debugger for parallel code, was used to identify potential runtime errors in the code. Vampir, a tool from Pallas GmbH that provides visualization and analysis of MPI resources, allowed me to analyze the parallel code to see the advantages of using MPI.

I also experimented with the Monte Carlo algorithm to learn to convert serial code to a parallel version using both MPI and Open MP. This enabled me to compare the difficulty level involved in transforming a program into each version. I also looked at the Numerical Aerodynamics Simulations (NAS) OpenMP Parallel LU Benchmark using AssureView, a versatile tool from KAI (now Intel) that can be used to identify and correct a wide variety of logical OpenMP errors, including deadlocks and race conditions.

Overall, I have had a dynamic and challenging internship that has exposed me to a variety of aspects in parallel computing. I have learned how to change serial code into parallel code using both MPI and OpenMP and have gained a better understanding of several tools used for debugging and tuning parallel applications. The massive amount of information that I have received from PET has enhanced my knowledge and skills with programming and computer systems.

I feel confident about what I have learned and plan to apply it to future endeavors. I have learned that parallel computing consists of architecture, libraries, directives, performance tools, and the ability to write multi-process and/or multi-threaded programs that produce correct results and make efficient use of parallel resources.

SITE-INDEPENDENT COMMANDS AT NAVO MSRC: A UCLI HOW-TO

Joe Werne, NorthWest Research Associates, Inc., Boulder, CO, and Jared Barousse, NAVO MSRC User Support

The Uniform Command Line Interface (UCLI) is a set of tools designed to provide a consistent set of commands and unified syntax across all of the Major Shared Resource Centers. Currently, the UCLI toolset consists of commands for accessing archive file systems and preparing and submitting jobs into a batch queue system. This toolset is also platform independent.

The NAVO MSRC supports the UCLI effort and has installed the most recent version on all its supercomputer and archival storage platforms.

In order to help users take advantage of the UCLI, this article includes a tutorial to demonstrate its use. An example script is included below that demonstrates most of what users need to do when running batch jobs using the UCLI's site-independent syntax.

ARCHIVE COMMAND

The archive command is designed to perform basic file-handling operations on the center's archival storage system. For example, to retrieve a file from the mass-storage system and place it in the current working directory, do the following:

```
archive get pathname
```

where 'pathname' is the path on the archival system to the file. If you need to retrieve more than one file, archive can do that too:

```
archive get -C path file1 file2 file3 ...
```

where the -C option identifies the 'path' locating the sequence of files'file1 file2 file3 ...' that you want to retrieve.

To learn more about archive's options and arguments, type 'man archive' on any NAVO supercomputing system or look at the online documentation at http://www.pstoolkit.org. A more detailed example is included below.

qprep COMMAND

The qprep command translates job-submission scripts from a platform-independent format to the local, platform-dependent format. This is useful when you run the same code on different queuing systems (either at different centers or simply on different platforms at the same center). The translated qprep script is written to a new file, which may optionally be submitted to a queue by qprep.

For example, the following script, named exampleScript, could be used on any DoD platform that supports the qprep and archive standards:

```
#!/bin/csh
#PSTQ job_name = run20.batch
#PSTQ stdout = run20.stdout
#PSTQ stderr = run20.stderr
#PSTQ mail =  e
#PSTQ mail_to = happyuser@anywhere.com
#PSTQ account = NAVOSLMA
#PSTQ parallel_env=parallel
#PSTQ queue = batch
#PSTQ wall_time = 12:00
#PSTQ nodes = 1
#PSTQ submit=y
#PSTQ END_OF_PREAMBLE


# define some useful parameters
    set executabe     = 'a.out'
    set massStorePath = 'KH/3D/R25003d/run20'
    set workDir       = '/work/happyuser/run20'
    set inputFiles    = (run20.dat0 run20.params)
    set outputFiles   = (run20.result1 run20.result2)


# change to working directory
    cd $workDir


# retrieve the input data to start the run
    archive get -C $massStorePath $inputFiles


# run executable
(Note: this syntax is specific to the IBM machine.)
```

```
    poe $executable


# move run output to archival storage, deleting
the local copy

    archive put -D -C $massStorePath $outputFiles


exit
```

Note that the script preamble uses the UCLI's Practical Supercomputing Toolkit (PST) syntax.

To submit this script to the local queuing system, you simply type:

```
qprep exampleScript
```

To learn more details related to qprep syntax, type 'man qprep' on any NAVO supercomputer platform or refer to the online documentation at http://www.navo.hpc.mil/us/FAQ/PS_Toolkit2.html.

New qprep and Archive Command Features

The UCLI is acquiring new features this year. So far the following methods are available for archive: put, get, ls, mkdir, and stat (to check the status of the archival storage system).

New methods currently being added to archive include: mv, rm, rmdir, chmod, and chgrp. Similarly, to enhance qprep's functionality, the following methods will be added to the UCLI:

☞ Jobrun - a method to define platform-independent syntax for running a parallel executable. In the example above, mpprun would be replaced with jobrun.

☞ Platform - a method to provide uniform queue naming, uniform work-directory name, uniform specification of the target computing platform, and uniform specification of the target queue system. This will allow users a glossary and syntax for dealing with these unnecessarily site-specific aspects of supercomputer center usage.

For more details on the UCLI To Do list, visit www.pstoolkit.org. If you would like to shape the future of the PST, please fill out a survey and let the developers know what you need. The survey is online at http://www.pstoolkit.org/Survey/index.html.

The new and improved User Support personnel at the NAVO MSRC. (L-R) Bryan Comstock, John Tippitt, Jared Barousse, Cheryl Dedeaux, Sheila Carbonette, Sheri Helman.

Navigator Tools and Tips

NAVO MSRC Help Desk FAQs

John Cazes, NAVO MSRC User Support

Do you get frustrated trying to find answers to those seemingly simple, yet elusive questions about the NAVO MSRC systems? You can be assured that you aren't alone. With so many environments in one place, it is hard enough to remember the machine names, much less every little quirk about each system.

Most people spend a lot of time looking for the answers to the very same questions. First, they search the NAVO MSRC website, and then an internet search engine...and only after running out of search keywords and patience do they call the Help Desk, where finally, the question is answered, and all is well.

The problem at this point is 45 minutes were spent trying to find out how much of your allocation you have used rather than working with the 200 Gigabyte (GB) data file your program has generated.

Our job here at the Help Desk is to solve problems for the user community in the shortest possible time, so that their important scientific work can move forward. With this in mind, we want to catch as many people at the very beginning of the search process and get the question answered right away.

We have gathered together as many common questions as possible into a new User Frequently Asked Questions (FAQ) section, accessible from the NAVO MSRC web page at http://www.navo.hpc.mil/faq.html. The FAQ is intended to be a handy quick-reference that shortens some of those 45-minute quests for information to something more reasonable, say, 45 seconds.

For example, here are two sample questions from the new FAQ:

Q: WHY DID THE COMPUTER ASK FOR MY PASSCODE TWICE, AND THEN FAIL TO LOG ME IN?

A: This is a symptom of repeated passcode mistakes while trying login: After two attempts, the system enters "Next Token Mode." What this means is that you must enter your regular passcode AND a verification passcode to gain access. When you get a second passcode prompt, you must allow the SecurID to cycle to the next random number and then you enter the next number into the second passcode prompt. On the second passcode, enter the number directly off the display—the second passcode will fail if you try entering your Personal Identification Number (PIN) a second time.

Many users get frustrated trying to get out of Next Token Mode. The Help Desk can immediately clear the Next Token Mode, so if you fail to get past it several times in succession, just call and we will assist you.

Q: HOW CAN I TELL WHAT VERSION COMPILER I AM USING?

A: In order to check the version information on the IBM compilers, you must use the "lslpp" command. To check the Fortran compiler, you would use "lslpp -l | grep xlf" and for the C/C++ compilers, you would use "lslpp -l | grep vac"(vac stands for VisualAge Compiler, which is the official IBM name of the C/C++ compiler set). Unfortunately, this produces more output than most users really want; there is no simple "-v" option available.

On the CRAY systems, the "-v" compiler option shows the default compiler version. In addition, the "module avail" command outputs all available versions of libraries and compilers.

For users who are new to our systems, the FAQ section also contains information and links to help get you oriented with the different systems. And remember, whether you are a new or experienced user, you can always call the Help Desk, and we will be happy to help you to avoid the frustration and get the problem solved.

NAVOCEANO MSRC analysts provide technical assistance Monday through Friday via the Help Desk telephone:

(800) 993-7677

(228) 688-7677

DSN: 828-4161

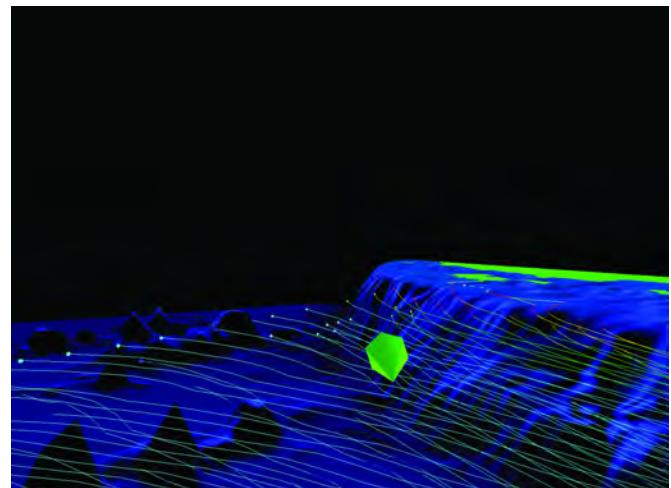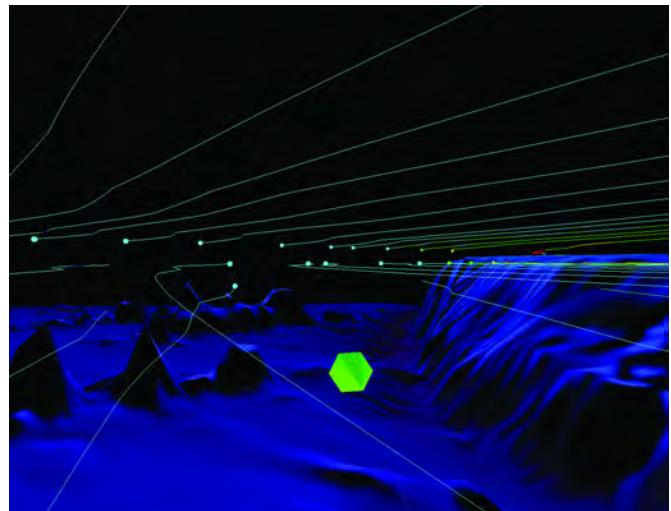or by email (msrchelp@navo.hpc.mil) during the prime shift from 0800 to 1630 CST.

CAVE Continued...

input devices as well as separate threads for each graphics pipe. VRJuggler also provides a cross-platform thread class for allowing developers to further divide computations onto more threads. By dedicating separate threads for data loading and visualization calculations, rendering threads are left only with the responsibility for sending rendering primitives down the graphics pipeline.

Another important issue with designing CAVE applications is the limited amount of input options. Most CAVE environments have a simple wand with 3 to 4 buttons and an optional joystick. Since users need to be able to move themselves around the virtual environment and be able to set visualization parameters and interact with the environment, it was necessary to design a virtual Graphical User Interface (GUI) complete with movable windows, buttons, checkboxes, and sliders. By designing the GUI in a modular fashion similar to most 2D GUIs, configuration panels could easily be created to allow the user to change settings such as which depth to slice the data or which data variable to color the massless particles.

CONCLUSION

The NAVO MSRC Visualization Center has the expertise to provide tailored applications to allow scientists to visualize their data. While many applications are designed for desktop workstations, through the use of tools such as VRJuggler and cooperation with universities such as MSU, the Visualization Center can also provide applications designed to run on the cutting edge of high-end virtual reality environments. The CAVE application designed for ARSC is one example of the type of solutions that can be provided. It is hoped that this application proves useful to the researchers at ARSC. The NAVO MSRC Visualization Center staff solicits any other users in need of this type of technology to contact them at viz@navo.hpc.mil.

Display of particle traces through an ocean model.

Coming Events

December 1-4, 2003	**Cluster 2003: IEEE International Conference on Clusters** Hong Kong www.csis.hku.hk/cluster2003
December 17-20, 2003	**10th Annual International Conference on High Performance Computing (HiPC 2003)** Hyderabad, India www.hipc.org
January 18-23, 2004	**2004 International Symposium on Collaborative Technologies and Systems (CTS'04)** San Diego, CA http://www.engr.udayton.edu/faculty/wsmari/cts04/cfp.html
February 25-27, 2004	**SIAM Conference on Parallel Processing for Scientific Computing (PP04)** San Francisco, CA www.siam.org/meetings/pp04
March 14-17, 2004	**PerCom 2004 - Second IEEE International Conference on Pervasive Computing and Communications** Orlando, FL www.percom.org
March 30-April 1, 2004	**HPCC'04 18th Annual HPC Conference** Newport, RI www.hpcc-usa.org
April 18-22, 2004	**Advanced Simulation Technologies Conference 2004 (ASTC'04)** Arlington, VA http://www.scs.org/confernc/astc/astc04/cfp/astc04.htm
April 18-22, 2004	**High Performance Computing Symposium 2004 (HPC 2004)** Arlington, VA http://www.eng.uci.edu/~jmeyer/hpc2004/
April 26-30, 2004	**IPDPS 2004 - International Parallel and Distributed Processing Symposium** Santa Fe, NM www.ipdps.org
June 4-6, 2004	**International Symposium on High Performance Distributed Computing (HPDC-13)** Honolulu, HI www.hpdc.org
June 28-30, 2004	**VECPAR2004 - 6th International Meeting on HPC for Computational Science** Valencia, Spain http://vecpar.fe.up.pt/2004

Naval Oceanographic Office * MAJOR SHARED RESOURCE CENTER

1002 Balch Boulevard   .   Stennis Space Center, Mississippi   .   39522